Chapter 5: Representation, Learning, and Recognition

The aim in this chapter is to discuss how the dynamics of coupled maps, with their intrinsically complex transformation of an input pattern, contribute to the forming of representations. Given a scheme for representation, which I have argued is plausible from both biological and classical pattern recognition viewpoints, an evolutionary learning process is outlined which is intended to achieve that representation.

A few concepts from formal language theory, Markov chains, and probabilistic finite state automata are used in the following discussion without formal introduction. I have included appendices outlining basic concepts in these fields.

CODING AND REPRESENTATION

In neuroscience and neural network theory, the words coding and encoding are used in a variety of senses, often different than their usage in communications engineering. In the latter, coding normally indicates that for some data stream, a set of p symbols is replaced with q symbols in an alternate, usually smaller alphabet – the code. This is the *encoding* process, undertaken for a variety of purposes including data compression, error detection and correction, or satisfaction of certain constraints imposed by the operating characteristics of the channel or the decoder. Codes are normally *invertible* by some decoding process, indicating that the initial stream can be recovered exactly.

In both natural and biological neural networks studies, representation is a more appropriate descriptor of the transformation process than coding. However, If the representation involves a reduction in dimension from an initial encoding of some input, but can reconstruct the original input, we can properly speak of it as an invertible code.

In biological studies, the term code is often used to indicate that an external observer is able to discriminate between a small set of stimuli by observing a spike train; the neuronal discharges are then thought to realize a one-to-one mapping from a set of input states to distinct output states. This fits nicely the hypothesis of neurons serving as detectors, sampling and coding a single variable, and has allowed a good deal of analysis of coding in low level vision. It is often less clear in what sense the stimulus relates to a *complete alphabet*[27], a set capable of representing all recognizable stimuli, and whether some downstream receiving neuron uses the spike rate as a code, even when some form of correlation exists between the rate and the stimulus. Moreover, the previous discussion on non-classical receptive fields should caution us on accepting that assumption without careful scrutiny, and to consider alternate interpretations of the spike trains.

---

[27] The exception is when the stimulus set is a complete basis set, such as the Walsh functions used by Richmond et al. . Then all possible input stimuli of a given resolution are covered; this is a desirable property.

For example, there are indications from the auditory system that neurons code more efficiently for natural stimuli, due to nonlinearities in their response which match the correlation structure of natural stimuli (Rieke, Bodnar et al. 1995). This indicates that the code is not a linear combination of some basis set.

PARTITION CELLS AND SYMBOLIC DYNAMICS

The idea of coding sketched in the previous section presupposes discrete symbols and alphabets. How are we to bridge the gap between the real valued states and complex temporal evolution of the logistic map units and the world of discrete symbols? The field of **symbolic dynamics** has been developed to address precisely this issue.

The orbit T of a map specifies a sequence of points in the phase space. Suppose that we quantize the phase space into a number of equally sized regions. Now, any point in phase space is identified with exactly one such region, known as a **partition cell.** These cells can have labels *corresponding* to an alphabet ***A.***

Every (partial) orbit T in this dynamical system can be considered as a sequence of elements or a **word** in the language ***A****. The set of all sequences generated by the system T, along with a given partitioning is a language ***L***.

This simple coarse-graining idea allows complex dynamical systems to operate in the symbolic realms handled by formal language systems, automata theory, and by neural networks utilizing local coding. As one example, symbolic dynamics gives an alternative "negative" procedure for defining a formal language: a language may be defined simply by enumerating forbidden words, i.e. sub-sequences that may not appear in the output of a dynamical system.

In neural networks operating within the localist, rate coding framework, *activation on a particular output unit* is typically interpreted as some form of coding. For high dimensional systems of coupled maps (model oscillators), the natural equivalent is to consider the *population (occupancy) of a partition cell* as an activation state. By normalizing this occupancy by the total number of cells, a distribution over all maps can be used as an output vector with real values. This allows even those sites inhabiting phase space intervals corresponding to *reduced* firing rates to play a role in coding.

Recall the distinction between Marr's concepts of the computational and algorithmic levels of description in a vision task. We can now see that a computational level process - the view interpolation and normalization strategy of Poggio and Edelman described earlier – can be remapped into a different algorithmic level, with correspondingly different assumptions about the underlying primitive physical elements. In the algorithm applied in this study, computation and coding arises from synchronization processes, resulting in coding across partition cells in the state space.

The underlying units of activity corresponding to each lattice sites are ensembles of neurons which cooperate to produce the synchronization process, at both local and distributed scales. Modulations in bifurcation and coupling, at the CML level of abstraction, arise from the interaction of underlying time varying micro-circuit parameters, or the influence of distributed modules whose rhythmic impulses affect these changes. To study the problem algorithmically, however, these underlying causes need not be specified in detail; the dynamics and state flows are sufficient to capture the essential operation.

FORMING REPRESENTATION SPACES WITH PARTITION CELLS

We have now covered the relevant material to introduce an essential feature distinguishing the approach to pattern recognition developed here from the other methods reviewed. *The dimensions in which representations are formed and distances between objects are computed are defined by partition cells in the phase space*. The representation of an object as a point in this space is based on some statistical measurement of the trajectories occupying particular subspaces. This is in contrast to dimensions corresponding to individual features as in classical statistical approaches, or associated with individual units as in feed-forward neural networks.

In particular, the present algorithm uses instantaneous statistics sampled after the dynamical evolution with two stages corresponding to two parameter sets. Alternative measurement strategies could be envisioned; for example, one might measure the overall residence time in particular partition cells.

SYMBOLIC DYNAMICS AND SYNCHRONIZATION: THE ROAD PROBLEM AND GENERALIZATIONS

At this point, I want to consider the problem to be solved by the second (synchronization) stage of the Soca network. As described earlier, this consists of sharpening the distribution (e.g. approaching synchronization into one or more clusters) over the partition cells so that *spatial configurations of initial conditions corresponding to objects in the same category* map to identical or very close distributions, taking the partition cells as dimensions of a metric space.

It seems that this problem structure corresponds to a generalization of an open problem in graph theory known as the *Road Problem* (Lind and Marcus 1995) . The general problem has been open since 1970, while some special cases have been solved (O'Brian 1981). Since progress on this problem might in turn enhance the prospects for more rigorous treatment of bounds on the pattern recognition capability of the Soca network, a brief description seems in order.

The Road Problem can be understood by a colorful analogy. A lost driver headed for Austin calls the auto club for directions; and happily following the directions (consisting of a sequence of intersections and turns), she reaches her destination. But the driver realizes she never told the auto club where her starting point was! This is only possible if the highway network were labeled in such a way that the same sequence of instructions would lead a driver from *any city* to Austin at the same time. Clearly, this is true only for a special subset of graphs and road labels. If a graph has such a structure, the sequence of labels leading from any state to the terminal state is known as the *synchronizing word* of the graph. The formal statement of the Road Problem is restricted to graphs with out degree two:

*If **G** is a directed graph with out-degree 2, is strongly connected, and is aperiodic, there is a road coloring (set of arc labels) with a synchronizing word.*

How is this related to the partial synchronization task? Suppose that the states resulting from the desynchronization stage of the network cover all k partition cells in the state space. If we could construct a state flow graph and transitions such that all states converge to a single partition cell at time t, we would have an exact correspondence to the road problem with $\binom{k}{4}$ colors. The states of the graph correspond to k partitions on the state space, with 4 neighbors in space requiring arc labels with $\binom{k}{4}$ colors.

However, convergence to a single state implies total synchronization, and both intuitively and experimentally we will find that it is difficult to distinguish initial conditions for totally or largely synchronized lattices. Everything looks alike for strong synchronization, and objects will tend to "collide". Instead, we need to generalize the problem to reach *a particular distribution over the partition cells at time t*, or at least to approach such a distribution within distance ε over the space defined by the partition cells.

If we want to avoid mismatches with these classifiers based on partial synchronization, what limits must be obtained. Let $dmin(c_1,c_2)$ be the minimum distance between the mean distributions of any two exemplars of all pairs of classes. Let ε_m be maximum distance of any view's signature distribution to the mean of its class distribution. To avoid collisions in a nearest neighbor search, the normalization process must produce

$$\varepsilon_m < dmin(c_1,c_2)/\ 2$$

An analogy for this generalized problem, consider this "All Industries Traveling Salesman's Regional Convention" problem. We want to have the salesman, distributed around a graph of cities, converge to some favorable set of cities; if we are convention organizers, we would like to optimize the distribution to match the conference hall size in each city. We cannot allow each industry to all choose the same set of cities.

If we wanted a single network with a labeling of arcs and synchronizing word which sends all salesmen in all industries to the correct city, we have a challenging problem; if we want this to occur in some bounded number of iteratations, the challenge increases. The simpler problem is to send all salesmen (lattice sites) in a particular industry (object) to a satisfactory set of cities (partition cells), while trying to handle the "overbooking" problem through rules limiting the conference size and encouraging many small conferences. Even this seems considerably more complex than the original road problem.

DYNAMICAL RECOGNIZERS AND PICTURE LANGUAGES

I have already briefly discussed the idea of a dynamical recognizer and mentioned that previous investigators have considered images as spatial extensions of formal languages. Languages can be described deterministically with grammars; in contrast, the productions of a language are often described with the tools of communications theory.

While the earlier discussion of dynamical recognizers, was strictly in terms of deterministic processing, the use of distributions for representation and decision process recalls the well developed field of *statistical language learning*. While perhaps not strictly necessary, I will describe the deterministic computation involved in representation and recognition in a statistical framework – it seems almost necessary for the purposes of gaining intuition, and it is possible that techniques from that field may be used to improve the present system.

Communications theory allows statistical characterizationof a stream in terms of probabilities of emitting words, or sub-blocks. Enlarging the *memory* or *anticipation* (e.g. including higher order statistics) window of a stream allows more accurate estimation of the probabilities. In a one dimensional data stream, **co-occurrence probabilities** between 2 adjacent symbols (digrams), 3 adjacent symbols, and generally n-blocks can be measured from the stream. A model of such a Markov process, using such higher order higher order probabilities, can generate streams with increasing fidelity to streams generated from a grammar. A large literature on construction of such chains for decision processes exists (Rabiner and Juang 1986); (Charniak 1993).

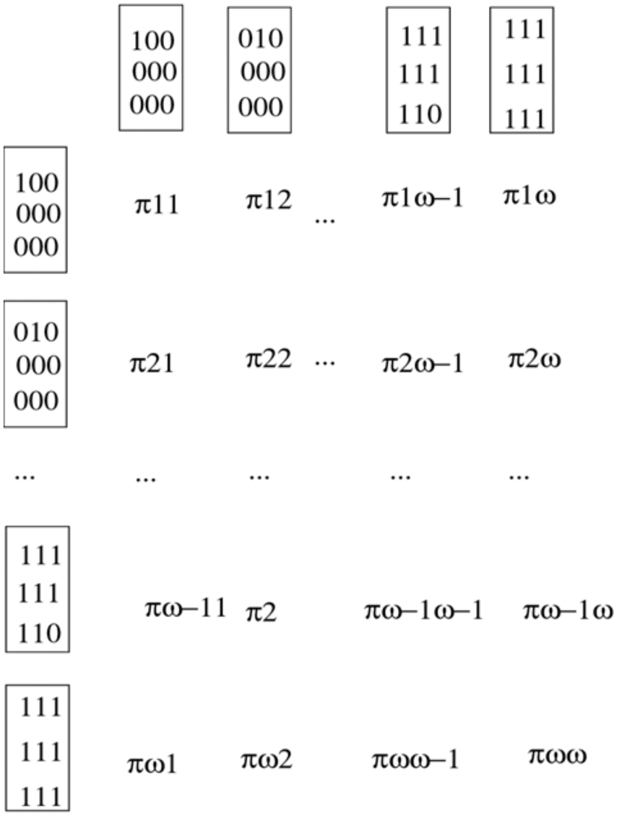

Fig. 22. A simple stochastic language description of an image or family of images. A matrix of blocks consisting of possible pixel configurations and probabilities that 2 such configurations are adjacent.

In a spatially extended system, the use of neighborhoods and iterations allows the recognition process to compute on the basis of a hierarchy of co-occurance statistics. After the first iteration, each cell (unit, pixel) has first order (2 block) information on the surrounding 4 on axis neighbors; after the second iteration, information from the overlapping 2 block statistics of surrounding 12 cells is included, after the third, information on the second order statistics of 24 cells is available, etc. The first order statistics condition the state flow of higher order statistics.

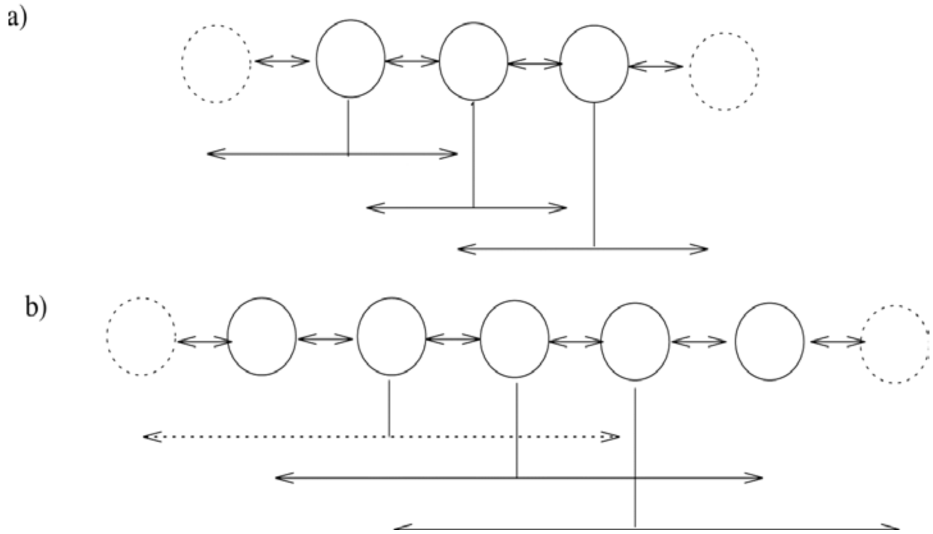

Fig. 23. The integration of higher order statistics over time. a) A one dimensional illustration of the increasing window size. After the first iteration, a 3-block has been analyzed. b) After the second iteration, each cell contains information about a 5-block, conditioned by the overlapping 3-blocks embedded in it. The result is a bottom-up interpretation of information at coarser spatial scales. Information integrated from the highest order statistics and spatial scales occurs in the synchronizing stage of the dynamics.

The end result of the dynamical evolution through two stages and coarse graining to k partitions is that each cell maps a point in a very high dimensional space to a lower one, limited by the precision required to encode the number of partition cells k. Since the images are given as binary images, each cell *after dynamical evolution* has indirect information about local statistics of various orders up to t1+t2, and the entire lattice has such information based on the original image in space of possible configurations with dimension 2^N. After iteration the value is quantized to a point in range 2^k. The occupancy statistics of all such cells are gathered to some precision, which need not be higher than $\log_2 N$, where N is the number of image pixels (and lattice cells). For values used in this study (k=64,N=5626) a dimension reduction of 49:1 is achievable for a family of 7 75x75 pixel images [28].

In the experimental work described later, the partition cell concept is only applied at the end of the dynamics, after sampling. However, in order to visualize how this dynamical system can *compute* similarity, it may help to consider the *intermediate* states as coarse grained, i.e. consisting of transitions between states defined by the

---

[28] The dimension reduction is computed as $7 * 75^2 / (64 * \log_2 75^2)$

partition cells. For chaotic dynamics, nearby points will by definition diverge, winding up in different partition cells. The resulting state transitions can be considered as similar to a *Markov chain*, where the transition probabilities between nodes corresponding to blocks of various sizes. The spatial dependence between cells and the deterministic dynamics result in the mapping being rather forced; it is unclear whether methods for constructing Markov chains from decision processes would be directly applicable.

The following diagrams illustrates schematically the processing of the Soca network serving as a classifier or recognizer of a picture, with the latter considered as a stochastic language. The network state flow is illustrated as a *nonstationary probabiliistic finite state automata* (PFSA), with each state corresponding to one of the partition cells.

The class learning problem consists of finding the two opponent stage parameters which label the edges and cells so as to map productions of the language (i.e. members of a class, projections of a 3D object) to the same distribution over the state transition graph at time t=t1+t2. Each arc is labeled with a transition probability to another state, when a site in that state is surrounded by an input word. The number of arcs between any two states is $\binom{k}{n}$ where *k* is the number of partition cells and *n* the number of neighbors.

The use of Markov chains on partition cells was recognized as a bridge between low dimensional dynamics and information processing models by Nicolis (Nicolis 1986) and Grassberger (Grassberger 1988); however, the Markov chain formalism cannot account for the spatial coupling of the CML. The extension to PFSA, by augmenting the transitions with *input words*, models the neighborhood-determined state transitions. (The use of an initial distribution of states is unusual for PFSA, but standard for Markov chains in distributed systems). To compute the next time step occupancy O in each partition cell i

$$O_i = \sum_{i=1}^{k} \sum_{w=1}^{w^*} \pi_{iw} O_{iw}$$

where k is the number of partitions, w is the word label on each arc, w* is the last of $\binom{k}{n}$ combinations for neighborhood size n, and π the probabilites for each input word state pair.

The Soca network parameters {*b1,c1,t1,b2,c2,t2*) and the number of partitions k form a concise specification of this PFSA graph. Of course, the two stages correspond to a relabeling of the transition probabilities, so the label on each arc is of the form

$$\begin{Bmatrix} \pi 1_{ij} w_m, t < t1 \\ \pi 2_{ij} w_{m,} t < t2 \end{Bmatrix}$$

While a PFSA model can be deduced from a CML, the two are not equivalent. The final state distributions computed by the two systems will diverge for images with equal first order statistics but differing in 2^{nd} or higher order statistics. This PFSA formalism is presented only for the purposes of visualization, as it is clearly an inexact mapping; it has less discrimination power than the actual CML. The actual CML is influenced by increasing order statistics (i.e. block adjencies) through time, while the PFSA over partition cells is limited to 1rst order spatial statistics at each step. The diagram on the following page illustrates the probabilistic, neighborhood dependent state flow between partition cells.

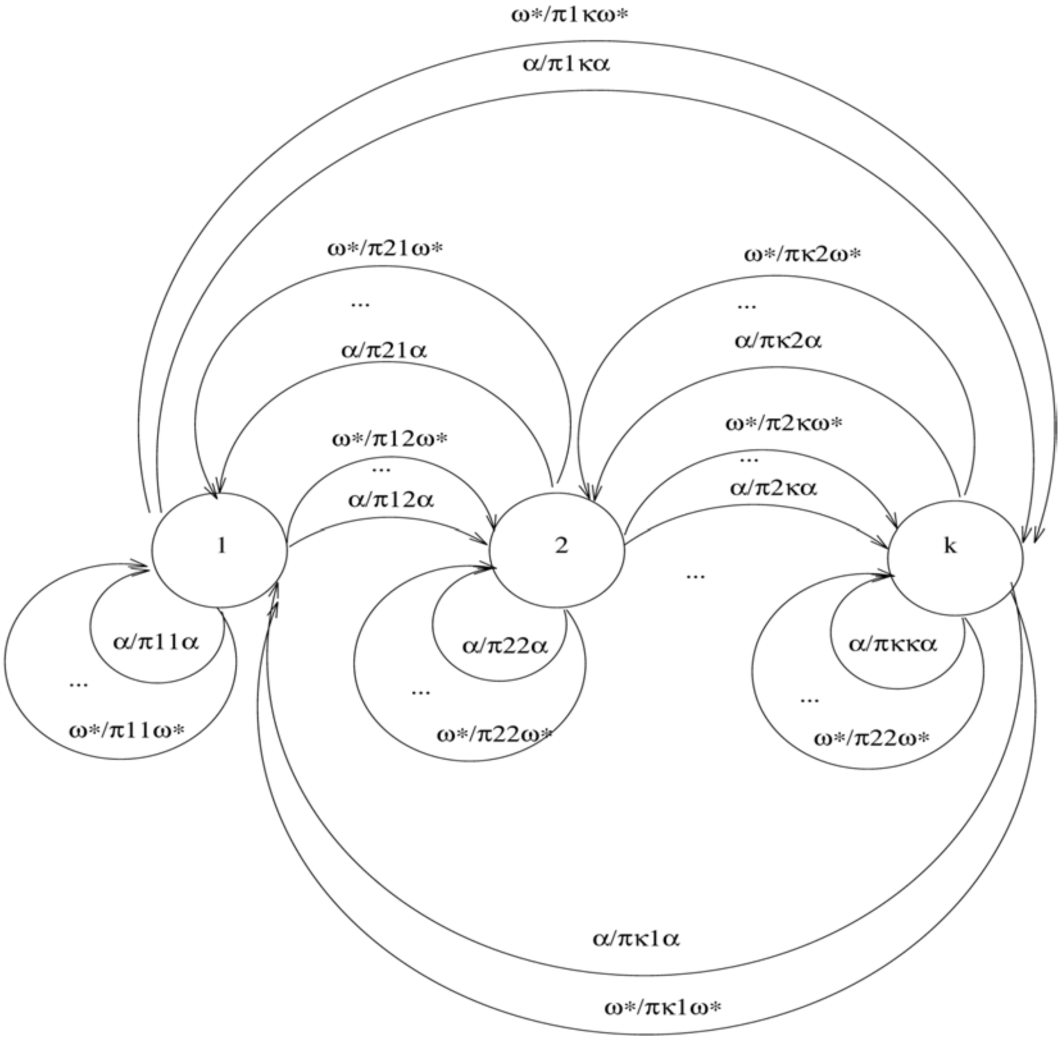

Fig. 24. The statistical operation of CML dynamics can be visualized as (and partially approximated by) a probablistic finite state automata. Each state in the PFSA corresponds to a particular partition cell, or interval of the phase space. Each arc is labeled by the probability of transition from source to sink state given the presence of an input word from alphabet {α,β, ... ω*}, Probabilities π for each input configuration and state are given on the arc; for simplicity, only one of the two stages in the synchronization opponent network is shown.

In summary, progress on mathematical frontiers including:

1. the computation of transient distributions for ensembles in coupled map lattices
2. coupling bounds leading to synchronization
3. the Road problem in automata theory

could result in the replacement of a heuristic search algorithm with a more direct solution, or to provable bounds on the power or limitations of the algorithms.

In the absence of such developments, improvements in the learning process beyond the present work might result from consideration of these issues. It is fairly easy to see, for example, that the state flow graph implicit in the second (synchronizing) stage must have paths of roughly equal length leading from the states occupied at the end of the first (desynchronizing) stage. It is less easy to see how to translate this to effective contraints on network parameters.

EVOLUTIONARY DESIGN OF SYNCHRONIZATION OPPONENT CLASSIFIERS

The mountain – Buddha's body
The torrent - his preaching.
Last night, eighty-four thousand poems.
How, how make them understand?

Sotoba, a Chinese layman (Translation by Lucien Stryk)

Evolutionary computing is a form of search intended to mimic aspects of biological evolution. A candidate solution to some problem is conceived of as a *phenotype*. A representation which can construct such a solution is the genotype, a collection of genes which code for individual parameters of the solution. Search takes place by forming a population of genotypes, evaluating the *fitness* of the phenotypes they specify according to some objective function, and letting the genotype material of the winners of this *selection* process assume a greater proportion of the population in subsequent generation. To reduce the likelihood of finding *locally optimal* solutions, the conservative action of selection is balanced by exploratory behavior in the form of mutation and recombination operators acting on the genotype.

Neural networks are generally specified by their topology, the activation functions of the units, connection strengths or weights, and any coordination or leaning structures (Balakrishnan and Honar 1995). Any or all of these may be subject to construction by an evolutionary learning process, but often known characteristics of the problem will constrain the network architecture. The learning process may fix some values, or may put constraints on some values (i.e. unit thresholds), while the problem specific values are set in specific.

Genotype representations represent an encoding of network parameters. This encoding may be direct, or indirect. A mapping of the genotype directly to weights in a fixed topology would be direct; a mapping of the genotype to a set of rewrite rules which must be recursively applied to specify a network would be indirect. The network here specifies bifurcation, coupling and number of iterations with a fixed, regular lattice interconnect topology; this is a clear example of direct encoding. In this example, the constraints on ranges that may appear in the genotype are set manually to rule out certain uninteresting parameter ranges.

Typically, the fitness function is a direct measure of the performance of the phenotype on the engineering task. In the present work, that approach is used in one toy problem, namely the search for parameters producing a quasi-metric space for a family of parametric curves.

However, for more complex tasks complete testing of candidate solutions may be impractical, at least for portions of the learning process, due to the computational expense associated with full testing. For example, the more challenging experimental task described in the next section is to identify particular objects from a set of similar objects, based on observing a single view after training on two or more views. Evaluating performance on the final task involves not only testing that all views of the object are recognized as the same object, but also that other objects are not mis-classified. Since the computation of the candidate solution is a relatively expensive nonlinear filtering operation, it is impractical to apply it to the full database of views, or even to a sub-sample to evaluate misclassification, during the evaluation of 3000 candidates phenotypes.

Instead, the performance on part of the task (recognizing different views of the same object) is emphasized, while certain *constraints on the output or encoding* produced by the network phenotype are incorporated into the objective function.

In addition, a reproduction strategy was chosen which preserves the best phenotype in each generation, guaranteeing a monotonically improving score.

Implementation and Performance Details

The current Matlab implementation of the evolutionary learning algorithm runs 30 generations of 100 genotypes. For the major object recognition task, learning the best representation for each object in the database (by using all 7 views) used 40-50 minutes/object on a 400 MHz PowerPC 750, 1M cache, 100 MHz system bus. The majority of time in learning (85%) is spent in the image convolutions of the diffusive step of the CML dynamics. The time variability results from branching around the computation of some objective function terms when a basic synchronization criteria is not met.

Each stage, or *attractor frame* consists of a triple $\{b, c, s\}$, where s is number of iterations in the stage. Two such frames are applied in turn. During evolutionary search for the parameter sets, the first stage is constrained to 2-6 iterations, the second constrained to 2-9 stages. In the evolved solutions, the total iterations required to create the representation space ranges from 6 to 14 iterations, with a mean of 11.3 iterations for the 39 objects in a 3D object recognition task. Each object in this set is 75x75 pixels, plus padding proportional to the number of iterations, to prevent array boundary diffusion statistics from contaminating the statistics. Thus each typical network phenotype evaluated represents roughly 77000 (40000+ 30000 * 11) multiply operations for the convolution and nonlinear map operations.

The simplex reproduction strategy (Karr 1991) governed the transmission of successful genes. It is well suited for situations where the time complexity of evaluating the gene is much greater than the processing associated with the evolutionary strategy, including complex image processing operators (Brigger 1995). In the simplex strategy, each adjacent pair of genotypes in the pool is evaluated as a local tournament, with the winner promoted to the next generation. The overall best-fit genotype in each generation is saved in element 1 of the genotype array, and is protected from mutation; this insures that the fitness is monotonically increasing and effectively makes the learning procedure a form of gradient descent. All winners of the pairwise tournaments are subjected to crossover mutation; given an effective rate of crossover of .5 for the whole population. The losing genotypes are replaced with copies of the best genotype from the last generation. These replacements are subject to mutation at rates given in the table in the experiments section.

The evolution parameters in these scenarios are the bifurcation parameters (b1, b2), the coupling parameters (c1, c2) and the iteration count of each of two dynamical synchronization opponent stages (t1, t2). Bifurcation and coupling are represented as doubles, the number of iterations as integers. The number of partitions k used to record the instantaneous occupancy at time t1 + t2 is also a critical value in the network performance; the values used in the work here were 256 for the curve evolution experiments and 64 for the object recognition experiments. This system parameter was not subjected to evolution, but held constant within a family of experiments.

Mutation was applied to the genotype copied from the best in generation to the pairwise competition losers, and to the winners after crossover. The mutation probabilities are set independently for each parameter as shown. Constraints on the evolution of parameters, as well as a general form of the desired solution (i.e. two dynamical “phases”), were provided as assumptions for this study.

The use of such constraints is perhaps unusual for evolutionary computing, and might be considered undesirable in the sense of biasing the solution. I argue that such constraints are justified here for at least two reasons. The general hypothesis being explored here was that a metric space could be created based on the dynamical flow, and solutions not conforming to this were not desired. Without such constraints – chiefly the lower bound on bifurcation parameter – solutions would rapidly emerge in the fixed point parameter regime which were clearly based on the size of the boundary. The high coupling regime, which can also lead rapidly to a fully synchronized fixed point, was similarly avoided. Such solutions could be avoided by adding terms to the fitness function; exactly this approach is taken in the next section, where excessive synchronization was found to degrade the performance of an object recognition system. Avoiding the regimes directly by restricting the parameters improves the search efficiency.

The learning algorithm is summarized below in pseudocode. The actual code uses an object-oriented control flow, but here I adopt a procedural style in the interests of clarity.

The variable *viewSet* is a set of related *images* (e.g. exemplars of a parametric curve, or different views of an object rotated in depth). The complete set of such families is the *objectWorld*. Variable *population* is an array of genotypes {b1,c1,t1,b2,c2,t2} and a field fitness to record the result of evaluating the fitness function.

```
procedure learndb
  for viewSet in objectWorld
    population = initializeGenotypeRandom;
    for generations = 1 to maxGenerations;
      generation(viewSet, population);
      population = simplexReproduction(population)
    end // generations
   objectDB(viewSet) = normalizedHist(lattice(viewSet, population(1))
   genotype(viewSet) = population(1);
 end  // viewSet
end learndb

procedure generation
   for all genotypes in population
         for each image in viewSet
          shapeDB(image)= normalizedHist(CML (genotype,image))
        end image
        genotype.fitness = fitness(shapeDB(image))
  end genotype
end generation
```

```
procedure simplexReproduction(population)
   bestfitGenotype =  find genotype with best fitness in population;
  if fitness of bestFitGenotype < currentBest, replace;
  strongHalf = winners of pairwise competition;
  apply crossover operator on winners;
  copy currentBest to losers;
   apply mutation to copies of currentBest;
   return population;
end simplexReproduction
```

Summary of Evolutionary Learning Method

A evolutionary learning framework for image processing problems was developed to train a two stage coupled-map lattice network using a computational strategy known as synchronization opponent cooperative action. The framework allows substitution of objective functions depending on the specific image processing task. Two tasks, described in the next part of the thesis, both require the formation of representation spaces whose dimensions correspond to partition cells in the network phase space. The objective functions differ slightly, however.

In *learning to order examples* from families of parametric curves, direct scoring on the position of points in the representation space is used. In *learning to identify an object* in a set, *direct evaluation* on a stimulus equivalence measure is combined with *indirect* evaluation of inter-object discrimination during the learning process. The representations emerge from these constraints, thus the learning should be considered as *unsupervised*.

The objective functions used vary depending on the exact task being performed, and are consequently described in the appropriate context in the following experimental work chapter; some details on the organization of Soca software package are given in the appendix.

An example of the evolutionary learning procedure for one task, the recognition of paperclip objects rotated in depth, is illustrated in the figure below:

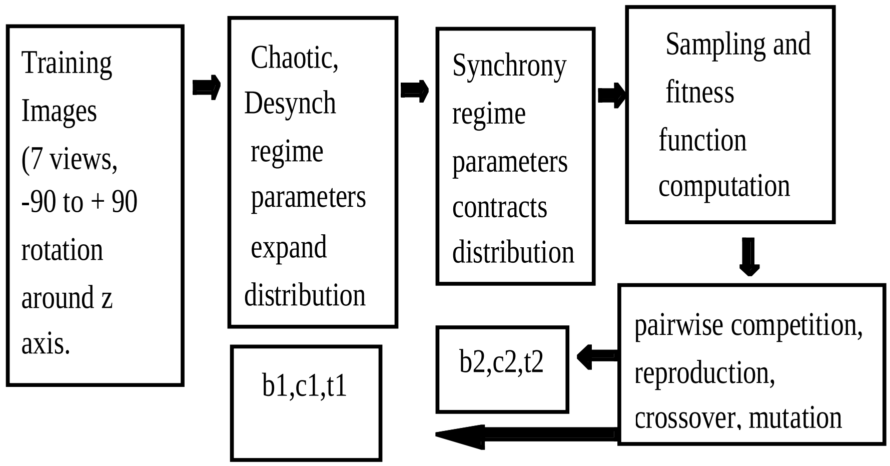
Evolutionary Learning in the Soca Network
Training Images (7 views, -90 to + 90 rotation around z axis.
Chaotic, Desynch regime parameters expand distribution
Synchrony regime parameters contracts distribution
Sampling and fitness function computation
pairwise competition, reproduction, crossover, mutation
b2,c2,t2
b1,c1,t1

Fig. 25. Binary images provide the initial condition for a network of recurrent logistic map units. The network is homogeneous in space, with bifurcation and coupling parameters between cells identical but time varying. The final result of training is a set of six parameters (genotype) and a distribution of state values obtained after a small number of time steps.

RECOGNITION WITH AN ENSEMBLE OF DYNAMICAL RECOGNIZERS

In this section, I describe in detail how the *recognition* process functions when the Soca network concept is applied to a pattern recognition task. The task and results are described in detail in the next chapter.

Recall the description of a dynamical recognizer given in the first chapter. A recurrent dynamical system was shown by Pollack to function, after training, as a recognizer for a language by reaching an accept or reject state after iterating over a bit string. In that case, the network functions much like a finite state automata. Each bit in the string is a bit in an input stream, with the next state computed from the input and the current state. Given an appropriate partitioning and labeling of the network's continuous state, the input output relation could be preserved and a finite state graph produced for a regular language.

The situation here is more complex, due to the higher dimensions and the 2-D spatial arrangement. Each cell in the CML functions like the one-dimensional recognizer in Pollack's scheme, with the following differences:

1. Rather than sequencing over sequence of raw states, at time t the cell receives a set of 4 inputs processed t-1 times by neighbors. Only cells in the first iteration receive raw input.
2. No cell can reach a decision on its own; the representation is distributed over the population of all units.

In Pollack's recognizer, when the last bit in a pattern is processed the recognition process is complete. Here, due to the parallel spatial processing, all bits are processed in the first iteration and every subsequent iteration; the number of iterations until recognition is given as part of input to the recognizer. In the Soca network, the *iteration counts are part of the genotype*. If recognition in a fixed number of time steps were desirable, this could have been given as a constraint, but the effectiveness of the search process under that constraint remains to be studied.

The nearest neighbor recognition process after creating a database for all of the objects, in pseudocode form, is as follows:

```
procedure bestMatch(image, genotypes, binMeans)
    for each genotype in database
      hist = dlattice(image, genotype);
      if (thisDistance(hist)  < bestDistance)
        return  index(genotype);
     else bestDistance = thisDistance;
end // bestMatch
```

In an alternative testing paradigm, two views of objects are presented; the algorithm applies the best match procedure as given above; if the same object is selected, the trial is a match, otherwise a no-match response. This allows the possibility of false positive and false negative judgements.